

Esercizio di coding con l'IA

Costruisci il progetto SupportDesk di questo modulo con ChatGPT o Claude partendo da un prompt già pronto che indirizza il modello alla specifica del modulo, poi rivedi, esegui ed estendi quello che ti restituisce — confrontandolo anche con la nostra build di riferimento su GitHub.



Questo progetto non lo scriverai a mano. Lo dirigerai.

Quasi nessuno scrive più codice Wisej.NET da un file vuoto: un assistente scrive la prima bozza e lo sviluppatore decide se è giusta. Funziona però solo se conosci la materia, ed è quello che ti ha dato il resto di questo modulo. Qui affidi quelle conoscenze a ChatGPT o a Claude, ti fai restituire il progetto del modulo come soluzione Visual Studio funzionante, e lo rivedi come rivedresti la pull request di un collega.

Che cosa stai chiedendo

Ottimizza il browser dei ticket di Support Desk: abilita il logging di EF Core in sviluppo, registra l'SQL e i tempi della ricerca non ottimizzata, sostituisci il caricamento delle entità con una proiezione, aggiungi AsNoTracking alle query delle griglie, aggiungi la paginazione con una dimensione di pagina di 50, sostituisci l'accesso lazy alle navigazioni con una proiezione o con Include, e documenta nelle note del lab i risultati misurati prima e dopo.

Il modulo chiede questi deliverable, e il prompt qui sotto li chiede all'assistente esattamente così:

- ✓ Logging di EF Core abilitato in sviluppo con l'SQL e i tempi non ottimizzati registrati
- ✓ Query della griglia riscritta come proiezione senza tracking
- ✓ Paginazione con dimensione di pagina di 50 e filtri indicizzati
- ✓ Nessun comportamento N+1 da lazy loading rimasto nel codice di visualizzazione
- ✓ Note del lab che documentano il miglioramento misurato prima e dopo

Perché è fatto così

Il browser dei ticket di Support Desk funzionava nel Modulo 3 con cinquanta ticket di prova. Con cinquantamila non funzionerà, a meno che la query non faccia meno lavoro e nel posto giusto. Il guasto è silenzioso: una griglia che impiega tre secondi perché il codice di formattazione ha toccato una proprietà di navigazione per ogni riga, una sessione che tiene migliaia di entità tracciate che nessuno modificherà, una ricerca che scansiona una tabella. Questo modulo parla di fare meno lavoro, in SQL, di proposito, e di dimostrarlo con un log invece che con un'impressione.

Questi sono i termini che l'assistente deve azzeccare. Se quello che ti restituisce li usa con leggerezza, non ha capito il modulo:

Termine	Che cosa significa qui
proiezione	Una Select che restituisce esattamente i campi che servono alla schermata, come TicketListItem, così il database unisce i nomi correlati in un'unica istruzione.
Include e Include filtrato	Caricamento eager delle entità correlate insieme alla radice; un Include filtrato restringe una collezione (gli ultimi cinque commenti) così un editor carica un aggregato controllato.
caricamento esplicito	Caricare una navigazione più tardi, su richiesta, tramite Collection o Reference dell'entry e LoadAsync, quando l'utente espande una sezione di dettaglio.
lazy loading e N+1	Proprietà di navigazione caricate in modo trasparente al primo accesso; in una griglia significa una query nascosta per ogni riga o per ogni accesso a una navigazione.
AsNoTracking e identity resolution	Risultati che scavalcano il change tracker per le schermate di sola lettura; AsNoTrackingWithIdentityResolution restituisce comunque una sola istanza per entità quando la stessa riga compare più volte.
paginazione a livello di query	Where, OrderBy, Skip, Take e Select composti prima diToListAsync, con un CountAsync separato per il totale.
logging di EF Core	L'SQL generato di ogni comando con la sua durata, tramite LogTo o Microsoft.Extensions.Logging.

Costruiscilo — scegli il tuo assistente

Entrambi i pulsanti aprono una nuova scheda con lo **stesso prompt già pronto**. Il prompt indirizza il modello alla specifica di questo modulo — un unico file Markdown con la lezione, la traccia del lab e i deliverable — così lavora sul materiale vero invece di tirare a indovinare che cosa sia il Modulo 6. Puoi aprire tu stesso [module-06.md](#) per vedere esattamente che cosa leggerà.



Usa ChatGPT

Apri [chatgpt.com](#) con il prompt di questo modulo già caricato. Funziona con GPT-5 o Codex; qualsiasi piano che possa navigare il web recupererà la specifica.



Usa Claude

Apri [claude.ai](#) con lo stesso prompt già caricato. Chi usa Claude Code può invece incollarlo in un terminale dentro una cartella vuota.



Mandalo così com'è e ottieni un progetto, non un frammento

Il prompt si chiude chiedendo l'intera soluzione in un unico `.zip` con i passi per compilarla ed eseguirla. Scaricalo, apri la soluzione in Visual Studio, premi F5 e SupportDesk si apre su `localhost`. Ti servono l'SDK .NET e il pacchetto NuGet `Wisej-4`; tutto il resto il prompt chiede all'assistente di scriverlo da sé.

Sto seguendo il Modulo 6 - "Caricamenti asincroni, dati correlati, filtri e prestazioni" - del corso LearnWisej "Data binding con EF Core", e voglio costruire con te il progetto del modulo. Rispondimi in italiano.

Comincia leggendo la specifica del modulo. È un unico file Markdown (in inglese) che contiene l'intera lezione, la traccia del lab e i deliverable:

<https://learnwisej.com/assets/courses/ef-core-binding/specs/module-06.md>

Poi costruisci un progetto Wisej.NET 4 completo che io possa aprire in Visual Studio:

1. Progetto "SupportDesk", C#, net8.0 o successivo, pacchetto NuGet Wisej-4 4.1.x, host Kestrel che chiama `app.UseWisej()`.
2. Costruisci quello che chiede il lab della specifica: Ottimizza il browser dei ticket di Support Desk: abilita il logging di EF Core in sviluppo, registra l'SQL e i tempi della ricerca non ottimizzata, sostituisci il caricamento delle entità con una proiezione, aggiungi `AsNoTracking` alle query delle griglie, aggiungi la paginazione con una dimensione di pagina di 50, sostituisci l'accesso lazy alle navigazioni con una proiezione o con `Include`, e documenta nelle note del lab i risultati misurati prima e dopo.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - Logging di EF Core abilitato in sviluppo con l'SQL e i tempi non ottimizzati registrati
 - Query della griglia riscritta come proiezione senza tracking
 - Paginazione con dimensione di pagina di 50 e filtri indicizzati
 - Nessun comportamento N+1 da lazy loading rimasto nel codice di visualizzazione
 - Note del lab che documentano il miglioramento misurato prima e dopo
4. Segui la specifica ovunque nomini una classe, una proprietà o un evento di Wisej.NET - usa esattamente quello. Dove lascia una scelta aperta, dimmi quale API hai scelto e perché.
5. Tieni l'autorità sul server: stato e validazione vivono in .NET, il browser disegna e riferisce. Non spostare mai una decisione in JavaScript solo perché lì è più comodo.
6. Rendi il risultato dimostrabile: devo poterlo eseguire e vedere il comportamento di cui parla questo modulo, compreso che cosa succede quando qualcosa va storto.

Prima di scrivere codice, riformula con parole tue che cosa costruirai e dove starà ogni pezzo, così posso correggere il progetto finché costa poco.

Quando il codice è pronto, impacchetta l'intera soluzione in un unico file .zip scaricabile e dammi i passi esatti per compilarla ed eseguirla.

Infine rispondi a queste domande della specifica, usando come prova la tua implementazione:

- Il tuo log ha mostrato 151 istruzioni per una schermata da 50 ticket. Quale riga di codice ha causato le 150 in più, e perché niente nell'applicazione ti ha avvisato?
- L'editor ha bisogno del ticket, del suo cliente e della sua categoria, e degli ultimi cinque commenti; il pannello dei commenti apre lo storico completo su richiesta. Quale strategia di caricamento serve ciascuna di queste tre esigenze, e perché la proiezione è la risposta sbagliata per l'editor?
- Un collega propone query compilate e pooling dei Context per il browser dei ticket prima di guardare il log. Che cosa misureresti per primo, e che cosa dovrebbero dire i numeri perché valga la pena fare una delle due modifiche?

Chiedimi qualsiasi cosa non ti sia chiara prima di cominciare.

Poi rivedi quello che ti è tornato indietro

Codice che compila non è ancora codice giusto. Le domande interessanti sono domande di progetto: è questo il controllo Wisej.NET che il compito richiede davvero, segue il ciclo di vita che il framework ti offre, lo stato autorevole sta sul server o è stato parcheggiato nel browser perché era comodo? Leggi il risultato da revisore, non da cliente.

Questo modulo ti dà la checklist. Rispondi a ciascuna domanda guardando il codice che hai appena ricevuto:

- ✓ Il tuo log ha mostrato 151 istruzioni per una schermata da 50 ticket. Quale riga di codice ha causato le 150 in più, e perché niente nell'applicazione ti ha avvisato?
- ✓ L'editor ha bisogno del ticket, del suo cliente e della sua categoria, e degli ultimi cinque commenti; il pannello dei commenti apre lo storico completo su richiesta. Quale strategia di caricamento serve ciascuna di queste tre esigenze, e perché la proiezione è la risposta sbagliata per l'editor?
- ✓ Un collega propone query compilate e pooling dei Context per il browser dei ticket prima di guardare il log. Che cosa misureresti per primo, e che cosa dovrebbero dire i numeri perché valga la pena fare una delle due modifiche?

Tutto ciò che hai dovuto correggere vale la pena di scriverlo. Una breve nota con le convenzioni e le decisioni a cui sei arrivato, incollata nel primo messaggio della sessione successiva, vale più di qualsiasi prompt ingegnoso.



Questo l'abbiamo già costruito — confronta il tuo con il nostro

L'implementazione di riferimento del Modulo 6 si trova nel repository pubblico degli esempi. Costruisci prima il tuo, poi apri il nostro: la parte interessante sono le differenze.



[Apri l'esempio del Modulo 6 su GitHub](#)

Oppure clona tutto con `git clone https://github.com/iceteagroup/LearnWisej-Samples` e apri `Data Binding with EF Core Course/Module 6`.



Il nostro esempio è un punto di partenza, non un prodotto finito

È stato costruito nello stesso modo in cui stai costruendo il tuo — con un prompt, poi corretto — e si ferma deliberatamente dove si ferma il modulo. Aspettati soluzioni provvisorie dove andrebbe codice di produzione, gestione degli errori essenziale e nessun test. Prendilo come un invito: usa quello che hai imparato in questo modulo, più l'assistente, per portarlo più avanti. Migliorare codice che non hai scritto tu è l'abilità che questo corso insegna davvero.

Tre prompt per andare oltre

Punta il tuo assistente sul progetto che hai appena costruito — o sul nostro e prova questi. Ognuno è un esercizio di revisione ed estensione, non una riscrittura.

- ✓ **Aggiungi il secondo caso.** *"Estendi questo progetto SupportDesk con un secondo caso che segue lo stesso schema del primo — un'altra entità, schermata o controllo della stessa forma, collegato allo stesso modo. Estrai in una classe base o in un servizio ciò che è davvero condiviso, ma solo dove se lo merita; due cose simili non chiedono sempre un'astrazione. Mostrami il diff e giustifica ogni cosa che hai estratto."*
- ✓ **Dimostra che funziona davvero.** *"Passa in rassegna questo progetto deliverable per deliverable rispetto alla specifica del modulo su <https://learnwisej.com/assets/courses/ef-core-binding/specs/module-06.md> e mostrami la prova che ciascuno è davvero fatto — il percorso nel codice, su che cosa cliccare e che cosa dovrei vedere. Dove un deliverable è soddisfatto solo in parte, dillo chiaramente invece di arrampicarti sugli specchi. Poi aggiungi quello che manca."*
- ✓ **Rivedilo come faresti con una pull request.** *"Accompagnami attraverso questo codice SupportDesk come se stessi rivedendo la pull request di un collega e non l'avessi scritto tu. Dove vive lo stato, che cosa succede dall'inizio alla fine su una singola azione dell'utente, che cosa rimanderesti indietro per modifiche, e quali parti non ti sentiresti tranquillo a mettere davanti a un cliente? Sii specifico e cita le righe di cui parli — non riassumermi il progetto."*



Portalo dentro il lab

Il lab pratico che chiude questo modulo chiede gli stessi deliverable che hai appena fatto produrre all'assistente. Usa quello che hai qui come forma della tua risposta e come prova a sostegno — non come la risposta in sé. Il lab viene valutato sul tuo ragionamento, e quella parte deve essere tua.